Architecture For *"Chicago"*

Ralph Lipe
Architect For *"Chicago"*
Microsoft Corporation

Agenda

- Design goals for "Chicago"
- General architecture via boot sequence
- New stuff:
  - Kernel, USER, GDI, networking
  - Registry, base, shell

Design Goals For Windows *"Chicago"*

- Support enhancements
  - New user interface, ease-of-use improvements
  - Plug and Play
  - Complete, protect-mode, 32-bit multitasking operating system
  - Network client, peer server
  - Win32® application interface

Design Goals For Windows *"Chicago"*

- Support market requirements
  - Compatible with MS-DOS®-based applications and drivers
  - Compatible with Windows 3.1-based applications and drivers
  - High performance
  - Small, runs as well as Windows 3.1 in 4 MB
  - Robust

Windows *"Chicago"* Boot Process - Real Mode

- Boot into modified version of real-mode MS-DOS - Winboot.Sys
- Mounts double-space partition and loads driver into high memory
- Determines hardware configuration
  - Used for multiconfig processing
  - May need to prompt user

Real-Mode Boot

- Winboot.Sys:
  - Processes config.sys (if exists)
  - Processes autoexec.bat (if exists)
  - Sensible defaults picked if none present
- Win.Com is automatically run
- Win.Com spawns DOS386.EXE, both exist for compatibility
- DOS386.EXE:
  - Allocates all initial memory

Now, The Interesting New Bits...

- Load Plug and Play drivers
- Initialize the I/O supervisor
  - Controls disk and SCSI adapter
- Load protected-mode file system
  - Take over FAT partitions
  - Take over double-space driver
  - Take over real-mode MSCDEX

Plug And Play

- Configuration manager begins process of device enumerations and driver loading
- Drivers dynamically loaded when their hardware is detected
- Same process happens at run time

I/O Supervisor

- Fully Plug and Play-enabled architecture
- Mini-port drivers are binary compatible with Windows NT
- Windows 3.1 “Fastdisk” style drivers are supported
- Takes control from real-mode drivers
- Provides a mapper for any devices that have a real-mode driver

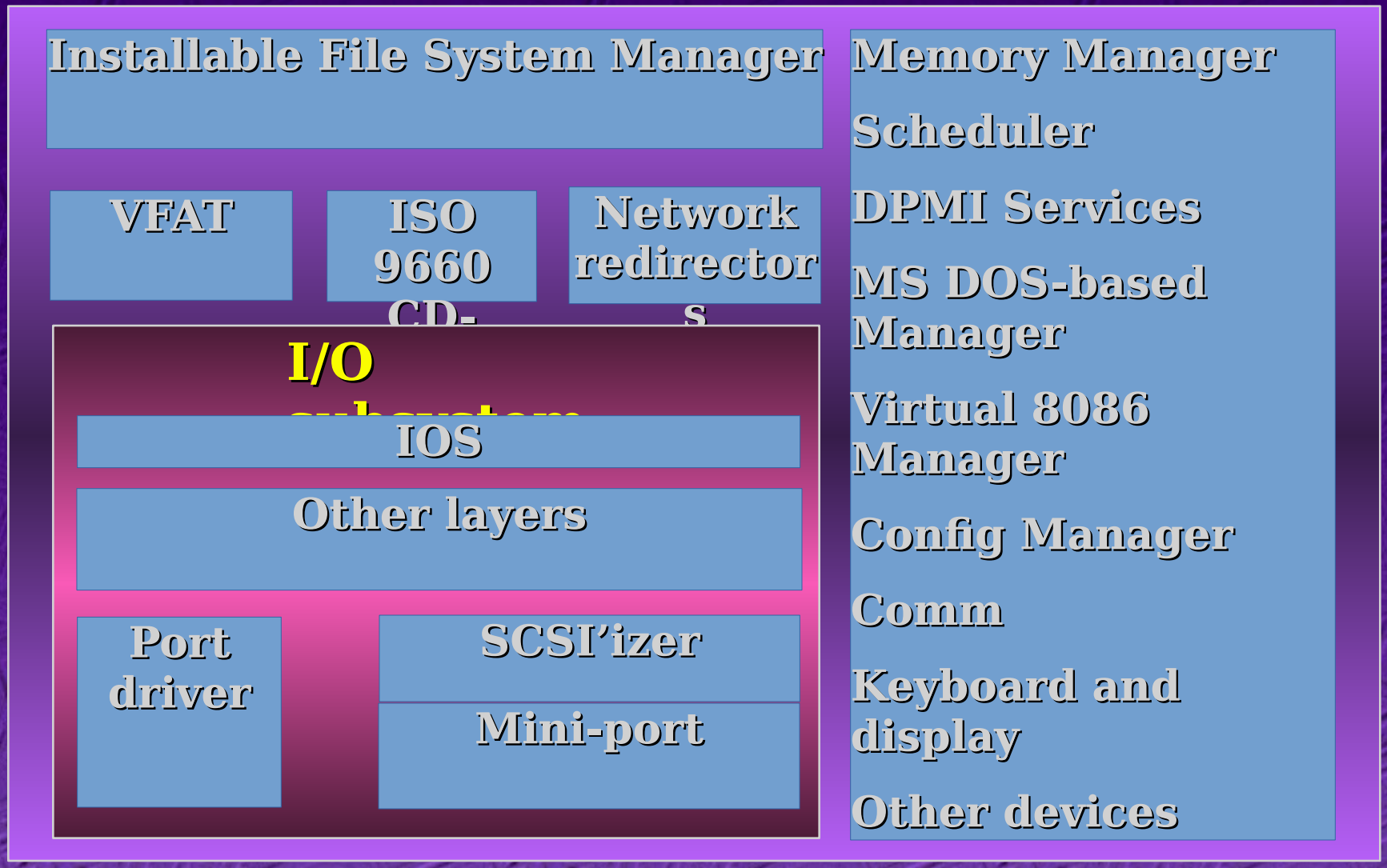
Installable File Systems

- Three file systems are provided
  - VFAT - controls local hard disk
  - CDFS - CD-ROM file system
  - Network redirector
- Each takes control from their real- mode counterpart (if one exists)
- All file systems support long filenames

Double Space Take-Over

- If any drive is double-spaced, the protected-mode driver takes over
- The MS-DOS memory allocated to the real-mode driver is freed

Ring 0 Components



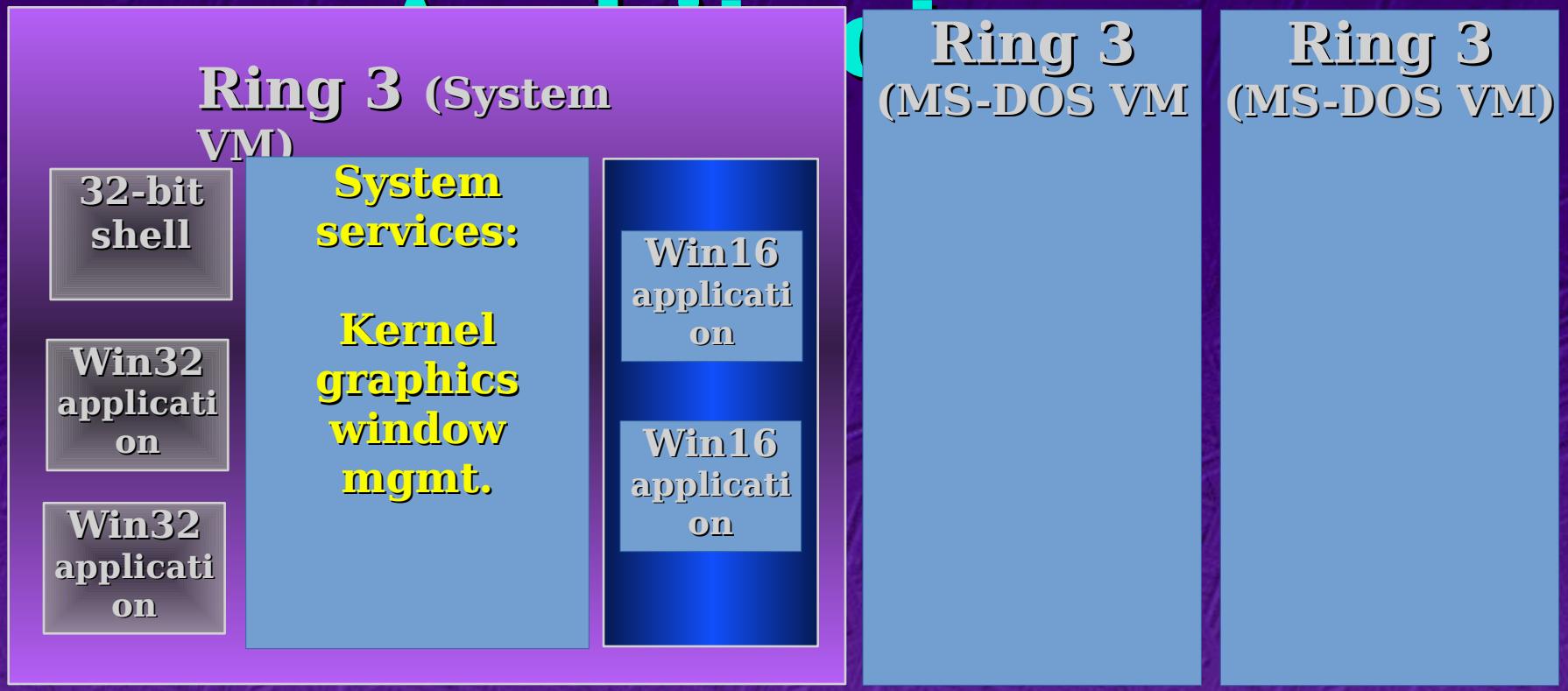
Windows GUI Initialization

- The Shell VxD executes Krnl386.Exe
- Windows kernel uses DPMI to switch to protected mode
- Loads GUI core components
  - Kernel/Kernel-32
  - USER/USER-32
  - GDI/GDI-32
  - Display, mouse, system, keyboard drivers
- USER runs the shell

Applications

- Every Windows-based application is run on a thread in the system VM
- 16-bit application must yield to let the next task run (just like Windows 3.x)
- 32-bit applications are preemptively multitasked
  - Asynchronous input model
  - Can be multithreaded

Windows "*Chicago*"



32-Bit GUI Components

- Every core component has two DLLs
  - New 32-bit interface
  - Same 16-bit interface for compatibility
- “Thunks” transfer control between 32-bit and 16-bit land
- Actual service may exist in 16-bit DLL, 32-bit DLL, or both
- Transparent to applications

Win32 API

- Windows NT™ binary compatibility PE file format (same EXE format)
- Lots of features:
  - Preemptive multitasking
  - Multithreading
  - Long filenames
  - Memorymapped files
  - Sparse memory
  - New shell APIs and controls
  - The list goes on and on...

New Stuff - Kernel

- Scheduler compatible to Windows NT
- Robust design:
  - Private address space for Win32 applications
  - Separate 32-bit fault handler thread for clean-up after application termination
- Extensible thunks
- Structured exception handling

New Stuff - USER

- Increased system capacity
  - 32-bit heap
  - Greatly increased limits on USER objects
- Robustness additions
  - Asynchronous input model
  - Ownership for all USER objects
- APIs for drawing 3-D windows and frame controls to support new shell
- Scalable window metrics for mobile users

New Stuff - GDI

- Improved resource limits
- Ownership for all GDI objects
- Image Color Matching
- 32-bit TrueType® rasterizer
- 32-bit printing subsystem
  - Metafile spooling
  - Bidirectional printing device support

New Stuff - GDI

- Paths and Beziers
- 32-bit DIB engine
  - Mini-display drivers
- On-the-fly resolution change

New Stuff - Networking

- Universal client
  - Multiple concurrent net clients
  - UI seamlessly integrated into shell
- Point and print
- Plug and Play-enabled card drivers
- Improved authentication and logon

New Stuff - Registry

- Support a subset of Windows NT registry API
- Registry intended to replace INI files
- Hierarchical database
- Can contain any data type
  - Text strings
  - Binary data
- Can be used by applications and drivers

New Stuff - Base

- Dynamic shrinkable swapfile
- Plug and Play driver support
- Dynamic VxD loading
- Param validation in debug VMM
- Improved event handling
- Improved block driver support
- Robust MS-DOS®-based application support

Call To Action: ISVs

- Write to Win32
- Write to OLE
- Use new Windows UI style
- Use new shell support
- Make applications aware of Plug and Play
- Smart setup

Call To Action: IHVs

- Fully Plug and Play-enabled
- Protect-mode device drivers
- Use “Chicago” mini-driver where appropriate
- Specific guidelines are in Windows Hardware Design Guide